Kapitola: Redukcie a porovnávanie zložitosti

1.1 Základná terminológia

Funkciu voláme *primitívne rekurzívna*, ak ju vieme vyrobiť z nuly, successora a projekcií pomocou konečného počtu operácií kompozície a primitívnej rekurzie.

Funkciu (potenciálne čiastočnú) voláme *čiastočne rekurzívna*, ak ju vieme vyrobiť z nuly, successora a projekcií pomocou konečného počtu operácií kompozície, primitívnej rekurzie a minimalizácie.

Funkciu voláme *rekurzívna*, ak je čiastočne rekurzívna a zároveň je totálna.

(Primitívne rekurzívne funkcie zodpovedajú programom len s for-cyklami. Čiastočne rekurzívne funkcie zodpovedajú všeobecným programom. Rekurzívne funkcie zodpovedajú tým programom, ktoré pre každý vstup zastanú.)

Nech $M \subseteq \mathbb{N}$. *Charakteristickou funkciou* množiny M nazveme funkciu

$$f_M(x) = \begin{cases} 1 & \leftarrow x \in M \\ 0 & \leftarrow x \notin M \end{cases}$$

Čiastočnou charakteristickou funkciou množiny M nazveme funkciu

$$f_M(x) = \begin{cases} 1 & \leftarrow x \in M \\ \perp & \leftarrow x \notin M \end{cases}$$

Množinu voláme primitívne rekurzívnou, ak jej charakteristická funkcia je primitívne rekurzívna.

Množinu voláme rekurzívnou, ak jej charakteristická funkcia je rekurzívna.

Množinu voláme rekurzívne vyčísliteľnou, ak jej **čiastočná** charakteristická funkcia je čiastočne rekurzívna.

(Rekurzívne množiny zodpovedajú triede rekurzívnych jazykov. Rekurzívne vyčísliteľné množiny zodpovedajú triede rekurzívne vyčísliteľných jazykov.)

1.2 Many-to-one redukcia

Hovoríme, že množina A *je m-redukovateľná* na množinu B (značíme $A \leq_m B$), ak existuje **rekurzívna** funkcia $f : \mathbb{N} \to \mathbb{N}$ taká, že $\forall n \in \mathbb{N} : \ n \in A \iff f(n) \in B$. Funkciu f voláme *many-to-one redukciou* A na B.

(Ak sa na množiny A a B dívame ako na rozhodovacie problémy, tak funkcia f predstavuje konečný algoritmus, ktorý zoberie ľubovoľnú inštanciu problému A a prerobí ju na inštanciu problému B tak, aby odpoveď zostala nezmenená.)

Existencia many-to-one redukcie nám v istom zmysle ukazuje, že problém príslušnosti do A je nanajvýš taký ťažký ako problém príslušnosti do B – totiž čokoľvek, čo vieme robiť s inštanciami problému B, vieme vďaka existencii f robiť aj s inštanciami problému A.

Relácia $\leq_m$ je reflexívna a tranzitívna, určuje nám teda tzv. kváziusporiadanie na množine $2^{\mathbb{N}}$.

Teraz môžeme definovať reláciu *m-ekvivalencie*: Množiny A a B sú m-ekvivalentné (značíme $A \equiv_m B$), ak $A \leq_m B$ a zároveň $B \leq_m A$.

Z reflexívnosti a tranzitívnosti $\leq_m$ a zo symetrie definície vyplýva, že $\equiv_m$ je skutočne reláciou ekvivalencie.

Každá many-to-one redukcia je vlastne program, ktorý vždy zastane, a programov je len spočítateľne veľa. Preto pre ľubovoľnú konkrétnu množinu A existuje len nanajvýš spočítateľne veľa množín B takých, že $A \leq_m B$. A z toho vyplýva, že každá trieda ekvivalencie relácie $\equiv_m$ obsahuje len nanajvýš spočítateľne veľa množín.

Všetky rekurzívne množiny okrem $\emptyset$ a $\mathbb{N}$ tvoria jednu triedu ekvivalencie v $\equiv_m$. (Totiž ak A a B sú netriviálne rekurzívne množiny, určite existuje f, ktorá o vstupe zistí, či patrí do A a podľa toho vráti buď konkrétny prvok z B, alebo konkrétny prvok z $\mathbb{N} - B$.)

Kapitola: Zložitosť rekurzívne vyčísliteľných množín

Vieme už, že pri rekurzívnej many-to-one redukcii sa nám trieda rekurzívnych množín rozpadne na tri triedy ekvivalencie – $\{\emptyset\}$, $\{\mathbb{N}\}$ a trieda obsahujúca všetky netriviálne rekurzívne množiny. Ako je to u rekurzívne vyčísliteľných množín?

Zatiaľ vieme len toľko, že všetky triedy ekvivalencie relácie $\equiv_m$ sú nanajvýš spočítateľne veľké, lebo pre konkrétnu množinu A prichádza do úvahy len spočítateľne veľa redukcii, a teda môže byť len spočítateľne veľa iných množín, ktoré sú s ňou ekvivalentné.

Na druhej strane, každá rekurzívne vyčísliteľná množina má svoju čiastočnú charakteristickú funkciu a tých je taktiež spočítateľne veľa.

Takže zatiaľ nič nestojí v ceste tomu, aby všetky rekurzívne vyčísliteľné množiny, ktoré nie sú rekurzívne, boli navzájom m-ekvivalentné, teda „rovnako ťažké".

Je to ale skutočne tak?

2.1 Kódovanie do čísel

Už v časti http://foja.dcs.fmph.uniba.sk/tvyp/skripta/05rec.pdf sme si ukázali, že primitívne rekurzívne funkcie vieme efektívne kódovať do prirodzených čísel. Hociktorý z uvedených postupov vieme zovšeobecniť pre čiastočne rekurzívne funkcie.

Z technických príčin sa nám bude viac hodiť, keď si samostatne očíslujeme **unárne** čiastočne rekurzívne funkcie. V celej tejto kapitole budeme teda uvažovať jedno ľubovoľné konkrétne efektívne číslovanie unárnych čiastočne rekurzívnych funkcií φ_n. (Teda φ_n bude unárna čiastočne rekurzívna funkcia číslo n.)

Rovnako, ako vieme očíslovať unárne čiastočne rekurzívne funkcie, vieme očíslovať všetky čiastočne rekurzívne funkcie s ľubovoľnou pevnou aritou. Rozmyslite si, ako očíslovať funkcie s aritou 0. Pre ľubovoľnú aritu $k > 0$ môžeme definovať číslovanie k-árnych čiastočne rekurzívnych funkcií napr. tak, že si zoberieme našu obľúbenú trojicu číslovacích funkcií c, l, r a definujeme, že $\varphi_n^{(k)}$ je funkcia, pre ktorú platí $\varphi_n^{(k)}(x_1,\dots,x_k) = \varphi_n(c(x_1, c(x_2, c(\dots, c(x_{k-1}, x_k)\dots))))$. Pre poriadok dodávame, že položíme $\varphi_n^{(1)} = \varphi_n$.

Teraz môžeme definovať množinu W_n ako definičný obor φ_n. Formálne, $W_n = \{x \mid \varphi_n(x) \neq \perp\}$.

Iný pohľad na to isté: W_n je množina, ktorej čiastočná charakteristická funkcia je $\chi_n(x) = \mathrm{sgn}(1 + \varphi_n(x))$. Keďže každá φ_n je čiastočne rekurzívna, je aj každá χ_n čiastočne rekurzívna, a teda každá W_n je rekurzívne vyčísliteľná.

A naopak, pre každú rekurzívne vyčísliteľnú množinu $W \subseteq \mathbb{N}$ je jej čiastočná charakteristická funkcia χ unárna a čiastočne rekurzívna, a teda existuje také n, že $\chi = \varphi_n$. Potom ale zjavne $W_n = W$. Preto postupnosť množín W_n obsahuje práve všetky rekurzívne vyčísliteľné podmnožiny $\mathbb{N}$.

Pre každú funkciu φ_n navyše existuje čiastočne rekurzívna funkcia g_n, ktorá má obor hodnôt W_n a navyše platí, že g_n je prostá a že ak $g_n(x) \neq \perp$, tak $\forall y < x:\ g_n(y) \neq \perp$. Takúto funkciu g_n voláme generátor množiny W_n. My si zvolíme g_n, ktorej program zostrojíme z programu pre φ_n nasledujúcou klasickou konštrukciou: Výpočet hodnoty $g_n(a)$ vyzerá tak, že paralelne simulujeme výpočty φ_n na všetkých x, a vždy, keď niektorý z nich skončí, zvýšime si počítadlo. V okamihu, keď počítadlo prekročí hodnotu a, tak aktuálnu hodnotu x vrátime na výstup. Je zjavné, že takto zostrojená g_n je čiastočne rekurzívna a má požadované vlastnosti.

2.2 Úplné množiny

Vieme o niektorej rekurzívne vyčísliteľnej množine povedať, že je najťažšia z nich všetkých? Ak áno, kedy?

Vtedy, keď je aspoň taká ťažká, ako hociktorá iná. A pre „aspoň taká ťažká" už máme jednu formálnu definíciu: reláciu $\leq_m$. Budeme teda hovoriť, že rekurzívne vyčísliteľná množina X je *úplná pri many-to-one redukcii*, skrátene *m-úplná*, ak pre každú rekurzívne vyčísliteľnú množinu Y platí $Y \leq_m X$.

(Poznámka: v literatúre sa m-úplné množiny niekedy tiež zvyknú označovať *kreatívne* množiny.)

Klasickým príkladom m-úplnej množiny je množina $HALT$ predstavujúca problém zastavenia.

Množinu $HALT$ definujeme nasledovne: $HALT = \{n \mid n \in W_n\}$. Slovne, $HALT$ je teda množina tých čísel čiastočne rekurzívnych množín, ktoré obsahujú samé seba – inými slovami, je to množina čísel tých unárnych čiastočne rekurzívnych funkcií, ktoré sú definovanú pre hodnotu rovnú ich číslu.

(To isté v reči programov: $HALT$ sú kódy tých programov, ktoré zastanú, ak na vstupe dostanú seba samého.)

Ak teraz zoberieme ľubovoľnú konkrétnu trojicu číslovacích funkcií c, l, r, môžeme analogicky definovať aj množinu $UNIV = \{c(x, y) \mid \varphi_x(y) > 0\}$, predstavujúcu univerzálny problém: ku každému programu x počítajúcemu unárnu funkciu zoberieme všetky vstupy y, pre ktoré zastane a vráti kladnú hodnotu, každú túto dvojicu (x, y) bijektívne zakódujeme do čísla a všetky tieto čísla tvoria množinu $UNIV$.

O množinách $UNIV$ a $HALT$ dokážeme, že sú m-úplné. Najskôr si ale uvedieme jednu technickú konštrukciu, ktorá nám zjednoduší život.

2.3 Veta o parametrizácii (s-m-n veta)

Veta: Pre každé $m, n \in \mathbb{N}$ existuje rekurzívna (a teda totálna) funkcia $s^m_n : \mathbb{N}^{m+1} \to \mathbb{N}$ s nasledujúcou vlastnosťou: pre ľubovoľný vstup $a, x_1, \dots, x_m$ nám funkcia s^m_n vráti taký výstup b, že platí:

$$\forall y_1, \dots, y_n : \ \varphi^{(n)}_b(y_1, \dots, y_n) = \varphi^{(m+n)}_a(x_1, \dots, x_m, y_1, \dots, y_n)$$

Ľudskou rečou, funkcia s^m_n transformuje programy. Na vstup jej dáme program A (zakódovaný ako číslo a), ktorý počíta $(m + n)$-árnu funkciu, a tiež hodnoty prvých m vstupov. To, čo nám funkcia s^m_n vyrobí, je program B (zakódovaný ako číslo b), ktorý už má len n vstupov a robí pre ne to isté, ako A s vstupom tvoreným zvolenými m hodnotami a následne tými n hodnotami, ktoré dostal ako vstup B.

Inými slovami, program B vznikne z programu A tak, že dosadíme za niekoľko prvých vstupov konkrétne, nami zvolené konštanty.

Dôkaz ľahko spravíme napr. pre náš „Pascal počítajúci funkcie“. Ukážeme si jeden názorný príklad, jeho zovšeobecnenie bude zjavné.

Uvažujme funkciu s^2_3. Tejto funkcii bude zodpovedať program S^2_3, ktorý robí nasledovné:

- Ako vstup dostane tri čísla: a, c_1, c_2.
- Z čísla a si zostrojí zdrojový kód programu A.
  BUNV nech hlavička hlavnej funkcie vyzerá nasledovne:
  `function main(x1,x2,x3,x4,x5 : integer) : integer;`
- Začneme vyrábať zdrojový kód programu B tak, že kopírujeme kód A, až kým nenarazíme na uvedenú hlavičku.
- V kóde A preskočíme uvedenú hlavičku.
- Do kódu B zapíšeme novú hlavičku `function main(x3,x4,x5 : integer) : integer;`.
- Skopírujeme z kódu A do kódu B definície lokálnych premenných a navyše pridáme `var x1,x2 : integer;`
- Skopírujeme zvyšok kódu A do kódu B, s jedinou zmenou: Do kódu programu B pridáme hneď na začiatok hlavnej funkcie príkazy `x1:=`c_1`; x2:=`c_2`;`.
- Skonvertujeme zdrojový kód B na číslo b a to vrátime na výstup.

Je zjavné, ako napísať takýto program, aj to, že na každom vstupe zastane.

2.4 Úplnosť problému zastavenia

Majme ľubovoľnú rekurzívne vyčísliteľnú množinu A. Aby sme dokázali úplnosť množiny $HALT$, potrebujeme ukázať, že $A \leq_m HALT$, teda že existuje rekurzívna funkcia f, ktorá „prekladá inštancie A na inštancie $HALT$".

Presnejšie, chceme ukázať, že existuje funkcia f, ktorá každé $a \in A$ preloží na číslo, ktoré patrí do $HALT$, zatiaľ čo každé $a \notin A$ preloží na číslo, ktoré do $HALT$ nepatrí.

Potrebujeme teda algoritmus, ktorý $a \in A$ prerobí na číslo unárnej čiastočne rekurzívnej funkcie, ktorá na svojom vstupe zastane, zatiaľ čo $a \notin A$ na číslo takej, ktorá na svojom vstupe nezastane.

Vyjdeme z predpokladu, že množina A je rekurzívne vyčísliteľná. Nech χ je jej čiastočná charakteristická funkcia. (Teda pre $x \in A$ je $\chi(x) = 1$ a pre ostatné x je $\chi(x) = \perp$.) Keďže χ je čiastočne rekurzívna, existuje n také, že $\chi = \varphi_n$.

Myšlienka toho, čo chceme spraviť, je jednoduchá: keď do χ dosadíme konkrétnu hodnotu a, výpočet χ skončí práve vtedy, keď $a \in A$. Bude tomu ale treba trochu doladiť technické detaily – potrebujeme z χ a a vyrobiť unárnu funkciu.

Keď poznáme $\chi = \varphi_n$, môžeme kompozíciou s P_1^2 vyrobiť funkciu $\chi^{(2)}(x, y) = \chi(x)$. Všimnime si teraz, že ak $a \in A$, tak $\chi^{(2)}(a, y)$ je pre každé y rovná 1, a v opačnom prípade nie je $\chi^{(2)}(a, y)$ pre žiadne y definovaná.

Nech m je číslo funkcie $\chi^{(2)}$, teda nech platí $\chi^{(2)} = \varphi_m^{(2)}$. Potom môžeme funkciu f definovať nasledovne: $f(x) = s_1^1(m, x)$, kde s_1^1 je rekurzívna funkcia z s-m-n vety.

Pre každé a je hodnota $f(a)$ rovná číslu unárnej čiastočne rekurzívnej funkcie $f_a(y) = \chi^{(2)}(a, y)$.

Ak teda $a \in A$, tak $f(a)$ je číslo funkcie, ktorá vždy zastane (a vráti 1), preto funkcia $\varphi_{f(a)}$ zastane aj na svojom vlastnom čísle, odkiaľ $f(a) \in HALT$.

A naopak, ak $a \notin A$, tak $f(a)$ je číslo funkcie, ktorá nikdy nezastane, preto nezastane ani na svojom vlastnom čísle, a teda $f(a) \notin HALT$.

Tým sme dokázali, že pre $a \in A$ platí $f(a) \in HALT$ a pre $a \notin A$ platí $f(a) \notin HALT$, a teda f je naozaj hľadanou redukciou.

2.5 Nemožnosť zúplnenia univerzálnej funkcie

Nech f je čiastočne rekurzívna funkcia. Hovoríme, že rekurzívna funkcia g je *rekurzívnym zúplnením* f, ak platí, že vždy, keď je f pre nejaký vstup definovaná, je preň definovaná aj g a vracia rovnakú hodnotu.

Všimnite si, že konkrétna funkcia f môže mať rekurzívnych zúplnení veľa. Napríklad pre funkciu $f(x) = \perp$ je dokonca každá rekurzívna funkcia jej rekurzívnym zúplnením.

Iný príklad: uvažujme funkciu $odm(x) = \sqrt{x}$, ktorej definičný obor je množina štvorcov prirodzených čísel. Rekurzívnym zúplnením tejto funkcie sú napríklad funkcie $f_1(x) = \lfloor \sqrt{x} \rfloor$ a $f_2(x) = \lceil \sqrt{x} \rceil$.

Definujme teraz funkciu $U(x, y) = \varphi_x(y)$. Táto funkcia U je univerzálnou funkciou pre triedu unárnych čiastočne rekurzívnych funkcií. Vieme už o nej, že je čiastočne rekurzívna. (Nie je na tom nič prekvapivé. U si z čísla x zostrojí program funkcie φ_x a tú následne simuluje na vstupe y.)

Vieme už, že definičný obor U je rekurzívne vyčísliteľná, ale nie rekurzívna množina. To preto, že nevieme rozhodnúť, či je funkcia φ_x na vstupe y definovaná. (Všimnite si, že definičný obor U zakódovaný pomocou párovacej funkcie c do prirodzených čísel je práve vyššie spomínaná množina $UNIV$.)

Položme si nasledujúcu otázku: má funkcia U nejaké rekurzívne zúplnenie?

Takéto rekurzívne zúplnenie U', ak by existovalo, by bolo celkom prakticky zaujímavé – dostali by sme simulátor ľubovoľnej funkcie, ktorý vždy v konečnom čase zastane a dá nám nejaký výstup. Samozrejme, tento výstup bude „nesprávny" pre tie prípady, kedy pôvodná funkcia nebola definovaná, ale to nám niekedy nemusí prekážať.

Inými slovami, na hodnotu $U'(x, y)$ sa môžeme pozerať ako na výrok „ak výpočet φ_x na vstupe y skončí, tak vráti hodnotu $U'(x, y)$".

Uvažujme napríklad známy otvorený problém: zistiť, či existuje nepárne dokonalé číslo (t. j. číslo, ktoré je rovné súčtu svojich deliteľov). Vieme napísať program, ktorý bude postupne testovať všetky nepárne čísla – avšak ak žiadne nepárne dokonalé číslo neexistuje, takýto program by bežal do nekonečna.

Ak by nejaká funkcia U' existovala, stačilo by nám opýtať sa jej na výstup nášho programu a následne overiť, či je to dokonalé číslo alebo nie. Mali by sme teda program, ktorý náš otvorený problém v konečnom čase vyrieši.

Ukážeme ale, že naše nádeje sú aj v tomto prípade márne, a teda že funkcia U žiadne rekurzívne zúplnenie nemá.

Sporom. Nech existuje konkrétna rekurzívna funkcia U', ktorá je zúplnením U. Použijeme štandardnú techniku – keď U' vie niečo povedať o každej čiastočne rekurzívnej funkcii, musí vedieť niečo povedať aj o sebe, a vďaka tomu vyrobíme funkciu, ktorá úmyselne vyrobí iný výstup ako si U' myslí.

Voľne podľa Scotta Aaronsona: Nič deterministické nikdy nemôže mať možnosť dokonalého sebapoznania – keby si vedel povedať, čo spravíš o desať sekúnd, mohol by si namiesto toho spraviť niečo iné.

In related news, práve vychádza prvá sezóna seriálu FlashForward, postaveného na veľmi podobnej premise: Celé ľudstvo na 137 sekúnd stratilo vedomie a počas tohto času každý videl svoju budúcnosť o pol roka. A hlavná otázka samozrejme je: stane sa naozaj o pol roka to, čo videli, alebo to práve vďaka tejto informácii dokážu zmeniť?[1]

Formálne, nech existuje rekurzívna funkcia U', ktorá je zúplnením U. Uvažujme teraz funkciu $f(x) = \overline{\text{sgn}}(U'(x,x))$. Keďže U' je rekurzívna, aj f je zjavne rekurzívna. A keďže f je navyše aj unárna, existuje n také, že $f = \varphi_n$.

A teraz už len počítajme: $f(n) = \overline{\text{sgn}}(U'(n,n)) = \overline{\text{sgn}}(\varphi_n(n)) = \overline{\text{sgn}}(f(n))$. (Jediná netriviálna je druhá rovnosť. Tá vyplýva z rekurzívnosti f – keďže f zastane, musí U' dať rovnakú odpoveď ako φ_n na vstupe n.)

No a už sme aj dostali spor – totiž funkcia $\overline{\text{sgn}}$ nemá pevný bod, a teda hodnoty na ľavej a pravej strane sú rôzne. Preto U' neexistuje.

Rovnako by sme vedeli dokázať, že funkcia $\text{sgn}(U(x,x))$ nemá rekurzívne zúplnenie.

2.6 Rekurzívne (ne)oddeliteľné množiny

Množiny A a B voláme *rekurzívne oddeliteľné*, ak existuje rekurzívna množina C taká, že $A \subseteq C$ a $B \subseteq \mathbb{N} - C$.

Intuícia: A aj B môžu byť zložité, nemusíme napr. vedieť rozhodovať, či $x \in A$. To, čo nám stačí, je algoritmus, ktorý vie rozlišovať medzi prvkami z A a z B. (A je nám jedno, ako sa tento algoritmus správa pre vstupy, ktoré nie sú ani z A, ani z B.)

Definujme množiny $FALSE = \{x \mid \varphi_x(x) = 0\}$ a $TRUE = \{x \mid \varphi_x(x) > 0\}$.

Tieto dve množiny sú zjavne disjunktné a rekurzívne vyčísliteľné. Nie sú ale rekurzívne oddeliteľné. Prečo?

Sporom, nech sú, nech C je rekurzívna množina, ktorá ich oddeľuje, a nech $TRUE \subseteq C$. Označme χ charakteristickú funkciu množiny C. Ľahko nahliadneme, že χ je zúplnením funkcie $\text{sgn}(U(x,x))$, čo je hľadaný spor.

(Poučenie: hranica medzi výpočtami, ktoré skončia a vrátia odpoveď „áno“ a tými, ktoré skončia a vrátia odpoveď „nie“ je natoľko zložitá, že ju nevieme rekurzívne počítať, ani za cenu toho, že môžeme dať ľubovoľnú odpoveď pre každý nekonečný výpočet.)

2.7 Jednoduché množiny

V tejto časti si zadefinujeme tzv. *jednoduché* množiny. Množinu voláme jednoduchá, ak je rekurzívne vyčísliteľná, má nekonečný komplement a tento komplement neobsahuje nekonečnú rekurzívne vyčísliteľnú podmnožinu.

Inými slovami, množinu voláme jednoduchá, ak je rekurzívne vyčísliteľná, má nekonečný komplement a zároveň má neprázdny prienik s každou nekonečnou rekurzívne vyčísliteľnou množinou.

[1] Je samozrejme jasné, ako to dopadne v seriáli, kvôli dramatickému efektu – naozaj sa stane to, čo videli, len to bude zasadené do nečakaného kontextu. Ale filozofická otázka je to pekná.

Zostrojíme teraz jednu jednoduchú množinu J. Pre každú množinu W_n dáme do J prvý prvok väčší ako $2n$, ktorý vygeneruje generátor g_n.

Formálne, nech $f(n) = \min\{x \mid g_n(x) > 2n\}$. Potom $J = \{g_n(f(n)) \mid f(n) \text{ je definovaná }\}$.

Oplatí sa všimnúť si, že do J *nedávame* najmenší prvok z W_n presahujúci $2n$ – tento totiž nemusíme vedieť nájsť!

Prečo je táto množina J jednoduchá?

Keďže pre každé n platí, že spomedzi $2n+1$ čísel $\{0, 1, \dots, 2n\}$ obsahuje J nanajvýš n, je zjavne komplement J nekonečný.

Nech W_n je nekonečná množina. Potom určite W_n obsahuje prvok väčší ako $2n$ (lebo ostatných je len konečne veľa). A teda nejaký takýto prvok skôr či neskôr náš generátor musí vygenerovať. No a prvý takto vygenerovaný prvok je v prieniku J a W_n.

A na záver, J je rekurzívne vyčísliteľná, lebo ju vieme generovať – stačí paralelne pospúšťať všetky generátory.

2.8 Jednoduché množiny nie sú m-úplné

V prvom rade si všimnime, že až také jednoduché zase nie sú. Žiadna jednoduchá množina nemôže byť rekurzívna. To by totiž bol rekurzívny aj jej komplement, a teda by obsahoval nekonečnú rekurzívne vyčísliteľnú podmnožinu – napríklad seba samého.

Na druhej strane však platí, že žiadna jednoduchá množina nie je m-úplná. A to aj vysvetľuje ich názov – toto boli prvé nerekurzívne ale rekurzívne vyčísliteľné množiny, o ktorých sa vedelo, že sú jednoduchšie ako m-úplné množiny.

Toto tvrdenie teraz dokážeme. Majme ľubovoľnú jednoduchú množinu A. Ukážeme, že predpoklad jej m-úplnosti vedie k sporu.

Uvažujme množiny $FALSE$ a $TRUE$ z predchádzajúcej časti. Množina $FALSE$ je rekurzívne vyčísliteľná, preto musí platiť $FALSE \leq_m A$. Nech f je rekurzívna funkcia zodpovedajúca tejto redukcii.

Funkcia f zobrazí prvky z $FALSE$ na prvky z A, a všetky ostatné prvky zobrazí na prvky z komplementu A. Keďže $FALSE$ a $TRUE$ sú disjunktné, všetky prvky z $TRUE$ zobrazí f na prvky z komplementu A.

Keďže $TRUE$ je rekurzívne vyčísliteľná, je rekurzívne vyčísliteľná aj $B = \{f(x) \mid x \in TRUE\}$.

Teraz sú dva možné prípady. Ak je množina B nekonečná, tak sme našli nekonečnú rekurzívne vyčísliteľnú množinu, ktorá je podmnožinou komplementu A. A toto je spor s tým, že A je jednoduchá.

A ak je množina B konečná, tak definujme množinu $C = \{ x \mid f(x) \in B \}$. Pre konečnú B je C zjavne rekurzívna a ľahko nahliadneme, že C separuje množiny $FALSE$ a $TRUE$, čo je opäť spor.